Subj:
Article from net.unix-wizards

Date:
Tue, 15 Oct 85 09:10:13 bst
To:
gdmr@ uk.ac.ed.ecsvax, rni@ uk.ac.ed.ecsvax, rwt@ uk.ac.ed.ecsvax,

gordon@ uk.ac.ed.ecsvax, fred@ uk.ac.ed.ecsvax, wex@ uk.ac.ed.ecsvax From: George D M Ross <gdmr@ uk.ac.ed.cstvax>

Re: editing over a packet net - response summary

Some months ago, I inquired as to what usenet readers know about the problems of editing under UN*X over a packet network. I should have posted this response much sooner, but two new systems which arrived here in the meantime (names of manufacturers will not be repeated, to protect the innocent) went into a time-hogging infant mortality mode.

If there's any further interest, mail me and I'll include your info in another summary; it won't take nearly as long as this one (Boy Scouts' honor).

Before I summarize, thanks to all who have communicated:
Rich Altmaier
hplabs!oliveb!ios!richa
Sean Casey
ames!vortex!cbosgd!ukma!sean
Gilles Chartrand
ihnp4!alberta!myriasb!ggc
Brad Davis
{ihnp4,decvax,seismo}!utah-cs!b-davis
R. Drew Davis
seismo!harvard!bentley!drew
Chuck Hedrick
seismo!topaz!hedrick
Eric Kolotyluk
seismo!ubc-vision!ubcg-mts!kolotyluk
604-228-4215
David Parter
seismo!uwvax!david
Jim Shankland
ucbvax!mtxinu!rtech!jas
Dave Sherman
ihnp4!utzoo!lsuc!dave
Maurice Suhre
ucbvax!trwrb!suhre
Chris Thomson
ihnp4!alberta!myriasb!cmt
Scott Weikart
scott@ Glacier.ARPA

Jim Wilson

201-631-7219
???

ihnp4!ihtnt!jhh
???

seismo!harvard!packard!hoxna!mdm

WHAT I HAD IN MIND

First, an explanation of the specific situations I had in mind is probably in order. I didn't make it clear as to whether I was thinking of one type of network, or was stuck with a network, or what. The primary networks which have to do with external access to this organization (NCAR) are (1) Uninet, a typical packet switched network which anyone can buy time on, and (2) remote ethernets connected to a local ethernet via TransLAN boxes over satellite links.

Uninet is probably typical of the major public packet nets. It has three basic algorithms allowed in the PAD (packet assembler/disassembler): (1) one character per packet, (2) all characters up to some terminator (e.g. newline) per packet, or (3) a time-slice arrangement in which all characters received within some amount of time (in units of 1/20 second, up to 255 units) are put into a packet. The problem with (1) is that it is expensive. Number (3) is a better match for UN*X editors, which typically issue a write() call on a tty line in raw or cbreak mode, but large writes (e.g. a total screen refresh) can end up being broken up into more packets than is really necessary. As pointed out by Maurice Suhre, Tymnet implements a slight variation on these tricks: it inserts a delay of 2.5 character times before sending out a character, to see if something else is on the way.

Regarding ethernets, this issue really concerns anyone who has an ethernet of any size (although it's of somewhat greater interest in a case such as the one here, where a demo system connecting remote ethernets, called USAN [University Satellite Network] is being put together in concert with a number of universities). It's obviously in the interest of your ethernet that you don't load it with oodles of 1-character packets, although that's actually what's going on if you're using any of the common terminal-server products to allow you to connect your terminals to multiple hosts. I don't believe any of the current products use terminator characters or time delays in an attempt to minimize the number of packets. (Can anyone correct me on that?) If that's the case, embarass your terminal server salesman the next time you see him; perhaps those companies will begin to improve their products in this area.

WHY DO THIS AT ALL?

I received several replies that questioned why anyone would want to do something like this, in the first place. There are several good reasons:

1)
People are already doing this, for instance, Eric Kolotyluk. Sites running IBM software with SNA are, in general, in a better position to support this

sort of thing than UN*X, since SNA can speak in packets.

2)
Line editing is only a substitute for screen editing if you're prepared to take the hit in productivity that implies. I personally don't know anyone who's using "emacs", "vi", or "e", who would care to revert to "ed". However, one user who replied says that he uses "qed" (a superset of "ed") for all his editing at 1200 (on a packet net) or at 19200, and that's a satisfactory solution for him.

3)
When running over a relatively slow (1200 or 2400 bps link), it isn't always practical to transfer a complete file to a remote site for editing. For instance, we have scientists who log in here to prepare jobs for submission to our Cray-1s, and if any of you know how many scientists program (no offense to the scientific community intended - after all, they're REAL scientists, not quiche-eating computer scientists :-)), you'd know that they often generate some very large, non-modular pogroms, er, programs.

4)
(mild flame on) Those of us who deal with UN*X frequently tend to forget that there are a few things it doesn't do well, and speaking in packets is one of them. We have to remember that there are lots of IBM system managers out there who will never go for anything else, but they sure won't even consider going for something else if that something else can't do everything they're already doing. (flame out)

5)
Economics: it's probably safe to say that slow links, i.e. 4KHz voice-grade channels, are going to be around for a lot longer than we'd like them to be.

A GENERAL SOLUTION

There are several things needed before a general solution to this problem emerges.

1)
There needs to be some sort of standard for packet communication between terminals and hosts. If this were the case, the packet wrapper would make communication between terminals and hosts truly transparent, doing away with the ugliness of escape sequences. Eric Kolotyluk mentioned that ISO/ANSI is working on a specification entitled "Basic Class Virtual Terminal Protocol" (document ISO/TC97/SC16N1276 - probably superseded by a later number), which would appear, at least from the title, to be addressing this terminal/host packet comm problem. Still, it is not likely that there will be any standards soon in this area, as pointed out by jhh:

1)
To make this work well and universally, a protocol is needed for standardizing communications between terminals and hosts. This will enable both terminal and host manufacturers to proceed with development of this software independently. This will also enable the editing features to be more controlled by the terminal rather than the host. One terminal manufacturer may provide an EMACS type interface, while another provides

a VI type interface. Since the same terminal would work on any host, the editing commands would not change from operating system to operating system, but from terminal to terminal.

This is a subject near and dear to my heart, as I work on the development of a packet switch network. To make packet switching cost effective, the data must be easily packetizable, but there is little effort to develop the protocols to packetize things until packet switching is more readily available, which won't happen until ... The old chicken and egg problem.

(2)
UN*X needs to learn how to speak in packets in a more general way. Sure, everyone running 4.2 has an ethernet, but I don't believe that can be called a general solution; there's always talk of the warts under the surface of 4.2 at Usenix conferences. I suspect that the "streams" capability being added to System V (D.M. Ritchie, "A Stream Input-Output System"; BSTJ, Oct. 84) is one promising approach to the problem of popping a protocol processing module into the path between a program and device; does anyone know of others?

(3)
There needs to be some sort of standard for splitting of functions between a host and an intelligent terminal/PC during editing sessions. Brad Davis pointed out that there's a document that provides some possibilities in this area, namely: Stallman, Richard, "The SUPDUP Local Editing Protocol" (MIT AI Lab, 6 Aug 84). (I've asked my friendly local librarian to get me a copy of this; it hasn't arrived yet, but if you want a copy, mail me and I'll send you one when I get it, IF it doesn't prohibit me from copying it.)

DISTRIBUTED EDITORS

Even without a generalized packet capability for UN*X, it's still possible to implement distributed editors, and several of those responding provided comments about the general characteristics of such an editor. There were two responses that described actual working editors, the first from jhh:

The jim editor for the Teletype DMD 5620 works pretty much the way you want. It still lacks some maturity and features compared with emacs or vi, and it is mouse based, for better or worse. It does have two parts - one running in the terminal, and one running in the host. The host handles such things as context searches, and the like, while the terminal handles the local editing, sending things back and forth in packets. A strategy where there is some intelligence in both the terminal and the host is needed. As the terminal-host connection is usually bandwidth limited, the minimazation amount of data transfer is desirable. This requires host software to determine what is the smallest amount of data that can be sent to the terminal, and terminal software to make small changes and send them to the host in a minimal amount of changes.

Even though jhh is talking about a specific product, his earlier comments about the need for standards are probably a good reason to believe that this distributed blit editor he mentioned won't be widely used because there is not a widely-recognized standard on which it's built.

The second response, from Rich Altmaier, was:

But on the general question, of what to do with a slower network, I believe use of a local processor ('pc') can work to advantage. The method here should not be some kludge but an editor designed to be split between local processor and host. My company built a product with a local processor doing editing, communicating with a host at 1200 baud. The local processor interacts with the host with a Remote Procedure Call model. A buffer of text is pulled into the local processor as needed. Changes are passed back to the host. Global operations such as text searches as passed to the host. The program written to do this is not all that complicated, given an RPC facility it is quite natural to design and write code knowing there is a boundary where control passes from machine to machine. Of course there are difficulties with data being duplicated, insuring host operations have the latest text image. We actually didn't use a pure RPC model, local changes are being sent to the host in parallel with further local editting. With this model, the host program consists of a set of procedures awaiting work requests from the local processor. The host program never interacts at the single character at a time level.

AN AT&T END-RUN SOLUTION

A couple respondents (R. Drew Davis, mdm) provided information about a solution that would be suitable if you're not already stuck with some net, and are shopping. This solution is in the form of the AT&T "Datakit VCS" product, which is said to provide low delay for character-per-packet traffic, even in the presence of other (large) packets. That is, you presumably don't worry about the inefficiencies of character-per-packet traffic; you just hook VCS up to your UN*X boxes and live happily ever after. >The primary drawback to this is that you have to buy a private net.

As I found from talking to Jim Wilson of AT&T Marketing (Morristown, NJ), VCS is not a tariffed product which is an alternative to public nets like Telenet. If you are prepared to pay for & install a private net, fine; you can get the benefits of their low ($_{10ms}$ node-to-node) delay, which is the result of special protocols and 8MB fiber LANs with 1.5MB (T1) wide-area links. But - there are currently no plans for the operating companies to offer it on a public-access basis like Telenet. And, there is always that annoying problem with the speed of light, which means they can't use satellites, which means you probably can't get this service to points overseas.

ETHERNET SOLUTIONS

Gilles Chartrand provided some interesting food for thought for those who have ethernets. Although his ideas are especially useful in an environment which links remote ethernets, they are also useful to those who have ethernets that are getting too crowded. The basic idea is this:

We have the same problem and have come the the following solution. In the case of vi, a temporary file is create on which the actual editing is done. Wich very small modifications this file can be made to exist on the CURRENT system. For example : If you are on machine A and you want to edit a file on machine B you would enter the following commands on machine A :

```
vi B:<file name>
```

The only time the network is used would be to transfer the file at the start and whenever a write is done and it would then be done in "block" mode for which packet networks shine!

.

.

.

We looked into modifying emacs and gave up before we could find any solid method.

It appears that getting the code to do this is easy:

In order to get the special vi, all you have to do is relink the .o files with the -lra (remote access) library. This package will allow you to prefix the files name with "<machine name>:". If no collon occures in the file name then vi (or whatever) works as before. Otherwise the file on the specified machine is opened via ethernet (like magic). There is a version of the csh which allows you to do things like "cd fred:" and presto, your on a new machine. "ls fred:/bin" will give you a listing of the files in your bin on the other machine. The remote access library and the remote csh are available from the Purdue University (E-mail path: ...!ihnp4!purdue!tichy) They are part of a larger project called IBIS which is destined to become a true, distributed file system.

Gilles also mentioned that work is being done at his site on a version of "rlogin" which, in normal tty driver mode, echoes remotely and doesn't send a packet until a timeout or a CR.

****

For those who have machines of the likes of Pyramids or Suns, I suspect that Sun's Network File System may provide a solution which is available now. Once a remote file system has been "mounted" on the local machine, it should be possible to cd, edit, etc. in the remote filesystem, although we don't know of anyone at this point who has been doing this on remotely connected ethernets. Do any of you know of such activity?

Ed Arnold * NCAR (Nat'l Center for Atmospheric Research)
PO Box 3000 * Boulder, CO 80307-3000 * 303-497-1253
csnet: erae ncar * arpa: erancare csnet-relay * uucp: ...!hao!scdpyr!era

Software Support Group Meeting

11.15am — 5th April 1984

Present: GB, JHB, GLC, HMD(Convenor), JGH, KMH, FK, GDMR, RWT

ACTIONS ARISING

Computing officer complement

Cover during KMH's absence:	
EMAS admininstration:	JHB
Filestore administration and consistency	JGH

General service support responsibilities.

Front-line management of APM service.	GLC
Improved APM Help information	all
User evaluation of APM services, esp Help	RNP
New VMS guide	vac student

Network Services

1. The upgrading of the VMS remote terminal access and file transfer is nearing completion, hindered by an elusive problem, and is to be documented for takeover.	GB
3. Some investigation of the development of XXX software for outgoing traffic is being undertaken.	JHB
4. Driving network printers directly was done before and could be done again without undue effort if the desirability were demonstrated.	none
5. The new Unix Vax can access the network and hence, rather indirectly, the VMS Vax. XXX might also be relevant. The pattern of likely activity is unclear and requires to be established.	The Unix community
6. Operational problems with the existing 2Mh ethernet need to be resolved; a better back-off algorithm will be introduced in the next release of ROMs; procedures for testing hardware need to be examined	RWT APM Review
7. A design study for the development of the 2Mh to 10Mh gateway and related matters is under way.	JHB
8. It was agreed to be desirable to link the VMS Vax to the ERCC Vax by 10Mh ethernet in order to gain experience.	JHB Tech staff

VMS login file modifications

Pruning proposals were put forward by GLC for comment.	all

IMP support

New IMP compiler and run-time library support on VMS Vax.	JGH

Archiving and backup

Consideration of improvements.	GDMR

Support for administration

DATATRIEVE is being evaluated.	GLC
Vac students and course co-ordinators were thought to be the only likely source of labour.	

Line-printer paper

80 or 132 columns on main printer.	80

Systems Discussion Group

I hope to make System Discussion Group Meetings a regular feature this year, with a meeting on each third Friday in the month. My intention is that interested parties should gather together for a detailed discussion of an active research topic. The October third Friday has been pre-empted by the VLSI Firbush trip, so the first meeting will be on the second Friday in October i.e. 11th at 3.00 in 3309. The topic is:-

WHAT SHALL WE DO WITH THE APMs?

The APMs are a valuable resource and clearly have an imporant and on-going role in the department. However, I believe they offer a valuable opportunity for us to make a major leap forward in the provision of an improved computing environment and to gain some credit in terms of publications. CMU proposed SPICE (Scientific Personal Interactive Computing Environment) some time ago, and I believe the APMs could be used in a similar manner. SPICE is an attempt to provide a personal computing environment as an alternative to a timesharing system, but with all the advantages of the latter in terms of inter-user communications and access to shared resources. They set off to use Perqs attached to an Ethernet; APMs must be better!

I would like us to discuss ways in which we should proceed and suggest we try to find answers to the following (and probably other questions):-

1. What would we expect such a system to look like?
2. What software (and possibly hardware) would be needed to create it?
3. How much research would be involved?
4. How much development would be involved?
5. Who would carry out the R & D?
6. How should it be funded (e.g. via SERC)?

Clearly most of the APMs must go on providing the service on which members of the department have come to rely. New R & D would need to take place on a designated sub-set although once a viable new system was established on the R & D set it could be introduced on to the rest in a controlled manner. This might indeed be an on-going process with several developments being introduced over a period of time. I have discussed these ideas with a number of you already and I know that one or two have written their thoughts down on paper. If we can get our ideas sorted out I believe we should get something in to the SERC for their December 15th deadline.

Please let Alison Fleming know as soon as possible whether you will be attending this meeting and whether you are likely to attend others.

RNI

From:
GDMR "George D M Ross" 28-OCT-1985 12:09
To:
NEWSYS
Subj:
Authorisation

Authorisation can't be combined with the storage of users' attributes unless we want to replicate the latter function. There will be more than one point of authorisation on the network, since there will be several "conventional" multi-access systems each of which is likely to want to do its own (terminal) authorisation. Any server on the network must be prepared to interrogate any of these authorisation points in order to find out who someone really is. (A "conventional" user of one of these multi-access systems is already identified and shouldn't have to "log in" again.) In exchange for identifying him/herself to an authorisation server the user receives a token which can be presented to any network server, thereby allowing the server to (a) decide which authorisation server issued the token, (b) decide if the network server trusts the authorisation server, and (c) ask the authorisation server who the token was issued to and where it may be used from. This scheme has the virtues that the user is only required to log on once in order to be acceptable to the entire network (assuming they logged on to a trusted host), and that clients aren't required to be trusted (most of them aren't!), only the authorisation servers.

From: GDMR "George D M Ross" 25-OCT-1985 13:59 To: RNI,RWT,FRED,WEX,GORDON,JHB,GDMR Subj: John's filestore document.

John's document seems to me to be based on the assumption that there will only be one kind of file server on the network, that being "The Filestore". I would contend that this is wrong. There will be a number of systems on the network which will be capable of acting as a file server. Some of these will be multi-access systems with local file storage, which export this (mainly) for the convenience of their "normal" users, while some will be dedicated machines (for cost/performance reasons). There may in fact be several varieties of each on the network.

We already have this situation, since ECSVAX exports its file storage capabilities to the rest of the (slow) ether. The code to do this was written because the old Interdata Filestore was too unreliable to support the development of the current filestores. It is fully capable of acting as a filestore, though it doesn't implement some of the more Filestore- specific features of the protocol.

So we really have two tasks. We have to design a file access protocol which can be used reasonably effectively to communicate with all the various systems which might want to export their file storage capability. We also have to design our stand-alone file servers. Since the various file systems on the network are already highly diverse, there is no reason why we should want to be artificially restrictive in the stand-alone servers just to avoid complication. The complication already exists, so we might as well design something functional rather than something primitive.

From:
GDMR "George D M Ross" 5-NOV-1985 16:30
To:
NEWSYS
Subj:
More on authorisation

In my previous suggestion authorisation tokens would only be valid from the particular protocol access point (whatever that is) to which they were issued. I did it this way because it doesn't rely on the communications media being unbuggable. Utilities exist to bug the current slow ether, for example.

However, things would be considerably simpler from the servers' points of view if the authorisation tokens had universal validity. A server would only have to validate the token, and not the means of access as well (this latter operation could get extremely messy if a number of protocols were in use). Of course, tokens would have to be difficult to forge, but I imagine that this could be done by including a suitably large "random" component.

It's not much use making the tokens unforgeable if they can be discovered merely by listening in to the communications media, though, so if we were to adopt this latter approach we would have to make sure that bugging was made difficult. For point-to-point links (logical or physical) that's OK. For the slow ether we can make life difficult by removing all the diagnostic capability from the stations of machines in untrusted areas. For the fast ether the only solution I can see is to put the controller onto the FRONT bus and to not allow the system components which talk to it to be replaced and to not allow anything else to talk to it directly either (unfortunately this rather goes against the philosophy of allowing the user to do anything at all to a personal machine).

Maybe this is an argument for not using 10 Meg ether. Maybe we should just revamp the slow ether. Maybe we should convert to something totally different (CentreNet? Something else?)

Comments, as usual, are invited....